An Adaptive Data Cleaning Scheme for Reducing False Negative reads in RFID Data Streams

Libe Valentine Massawe and Herman Vermaak
School Electrical and Computer Engineering
Central University of Technology
Bloemfontein 9301, South Africa
207071675@stud.cut.ac.za, hvermaak@cut.ac.za

Johnson D. M. Kinyua
Department of Information Technology
University of Northern Virginia
Annandale, VA 22003, USA
jkinyua.faculty@unva.edu

***Abstract*— Due to the high sensitivity of RFID tag-reader performance to the operating environment, RFID data streams generated are unreliable and contain a significant amount of missed readings. RFID data cleaning is therefore an essential task for successful deployment of RFID systems. One of the common techniques used by RFID middleware systems to compensate for the missed readings is the use of sliding-window filters. However, setting an optimum window size is non-trivial task especially in mobile tag environments. In this paper we present a new adaptive data cleaning scheme called WSTD based on some of the concepts proposed in SMURF but with an improved transition detection mechanism. WSTD uses the comparison of the two window sub-range observations or estimated tag counts to detect when transitions occur within a window. In the mobile environment, our experimental results show that the WSTD scheme performs better than SMURF producing an improvement of about 30% less overall errors than that produced by SMURF.**

Keywords-RFID; data filtering; data cleaning; WSTD; SMURF; RFID Middleware; sliding window filter

I. INTRODUCTION

Radio Frequency Identification (RFID) is used in a diversity of applications such as: distribution logistics, pharmaceutical and healthcare, library, contactless ID cards and tickets, asset management, manufacturing, garment industry, automotive industry, animal identification, traffic applications, aviation industry, military and many more [1, 2].

Although the performance of UHF passive RFID based system improved significantly by introduction of EPC Class-1 Generation-2 protocol (C1G2) [3], several studies on the performance of the C1G2 RFID systems indicates that the overall performance of the system is still implementation dependent [4, 5, 6, 7]. The empirical study of UHF RFID performance by Buettner et al. [5] shows that physical effects such as errors and multipath to be significant factors that degrade the overall performance of commercial readers. These effects increase both the duration of each reader cycle and the number of cycles to read all tags in a tag set. They argue that the error rates are highly location dependent and the level of degradation is implementation specific. The work by Kawakita et al. [7] shows that the bit errors due to erroneous communication link significantly degrade C1G2 performance. In actual UHF passive RFID deployment, the RFID's usually share the frequency band with other UHF wireless devices as well as neighbor RFIDs. While some interference is predictable and controllable some are unpredictable and uncontrollable such as mobile wireless devices. Therefore, despite the improvements on tag detection rates by using C1G2 protocol, factors such as tag-reader configurations, multipath and unpredictable interferences in the deployment environment still contribute to degradation of the performance and reliability of the RFID system leading to noisy and incomplete data.

RFID data cleaning is therefore essential in order to correct the reading errors, and to allow these data streams to be used to make correct interpretations and analysis of the physical world they are representing. RFID middleware systems are typically deployed between readers and application(s) in order to correct for dropped readings and provide clean RFID readings to the application logic [8]. This is an integral part of our ongoing work on developing multi-agent based RFID middleware system [9]. WSTD is used as a data cleaning mechanism for low-level RFID data processing tasks within our middleware system.

The remainder of this paper is structured as follows. Section II reviews the work on different methods for improving the reliability of RFID data streams by other researchers. Section III describes the statistical sampling perspective of RFID Data streams. Section IV describes the proposed WSTD algorithm to efficiently detect transitions. Performance evaluation and comparison of WSTD with SMURF and other fixed temporal-based sliding window schemes is discussed in section V. The conclusions and future work are given in section VI.

II. RELATED WORK

Methodologies for improving reliability of RFID data proposed in the literature can be divided into three main categories: physical solutions, middleware solutions and deferred solutions [10]. Physical solutions include improvement of hardware performance to improve the reliability of the data such as [11] and use of redundant techniques by using multiple tags and readers to identify the same object [12, 13]. Middleware solutions include algorithms to correct the incoming sensor data streams before the data is passes into the database [8, 14, 15]. The deferred solutions incorporate intelligent techniques which correct the data in the

later stages within the data storage [10, 16]. Our work falls in the category of middleware based solutions specifically window based smoothing methods. We decide to use window based method because of their simplicity and our work extends the work proposed by Jeffrey et al. [8].

Many commercial RFID middleware solutions [17, 18] contain a fixed temporal-based sliding window data smoothing filter as a solution to RFID unreliability, and applications are required to set the window size. Jeffery et al. [8] show that setting the smoothing window size is non-trivial task. It requires a careful balance between two opposing application requirements which are: (1) to *ensuring completeness* for the set of tag readings due to tag-reader system unreliability, and (2) to *capture tag dynamics* due to the tag movement in and out of reader's detection region. Large window sizes are good in ensuring completeness by smoothing out the missed readings but they are not efficient in detecting tag transitions.

On the other hand, small window sizes are able to detect transitions but they are not capable of compensating for the missed readings. Small windows lead to *false negative* errors in which the tag is mistaken assumed to be absent while it is actually present. In mobile tag environments big window sizes while trying to compensate for the missed readings introduce other errors which are known as *false positive* errors. False positive errors are readings in which the tags are mistaken assumed to be present while they have already exited the detection region. False positive readings are caused by interpolation of readings within big window sizes.

Taking into consideration the sensitivity of the tag-reader performance on the environment, it means that a small change in the environment can render the window unable to smooth the data. Jeffery et al. [8] proposed an adaptive sliding window cleaning method called SMURF (Statistical sMoothing for Unreliable RFid data). SMURF models the unreliability of RFID readings by viewing RFID streams as a statistical sample of tags in the physical world, and exploits techniques grounded in sampling theory to drive its cleaning processes. SMURF does not expose the smoothing window parameter to the application; instead it determines the most appropriate window size automatically and continuously adapts it over the lifetime of the system based on observed readings. We adopt the concepts proposed in SMURF and devise a cleaning scheme called WSTD with more efficient transition detection mechanism.

III. Statistical Sampling of RFID Data Streams

According to the RFID reader-tag performance analysis presented in [4, 6, 7], the raw RFID data streams do not provide a correct representation of the physical world which they are representing. A significant number of tags which are within the reader's read range are not consistently read by the reader due to either their orientation with respect to reader, distance from the reader, presence of metal, dielectric or water material close to the tag and other factors. These missing tags imply that typically only a *subset* of the tag population is actually observed on every read cycle. Hence, the observed RFID readings can be viewed as a random sample of the population of tags in the physical world. The key insight is viewing each read cycle output as a random sampling trial and the smoothing window as repeated random sampling trials. In our work we will refer to an atomic unit of time used by one read cycle as an epoch.

Let N_t denote the unknown size of the underlying tag population at epoch t and let $S_t \subseteq \{1, \dots, N_t\}$ denote the subset of the tags observed ("sampled") during that epoch. S_t can be viewed as unequal probability random sample of the tag population. Probability $p_{i,t}$ of selecting tag i at epoch t can be calculated from the epoch t output information using the number of reads (tag responses) for tag i in combination with the known number of interrogation cycles (number of requests) *a*s in (1)

$$p_{i,t} = \frac{number\ of\ responses}{number\ of\ requests} \quad (1)$$

IV. Window Sub-range Transition Detection (WSTD)

Building on the work done in SMURF, we developed our adaptive cleaning scheme for RFID data streams. Our scheme, called the Window Sub-range Transition Detection (WSTD) algorithm, uses binomial sampling concepts to calculate the adaptive window size and π-estimator to estimate the number of tags as proposed by SMURF. The main difference between SMURF and WSTD is related to transition detection mechanism. We used the comparison of the two window sub-range observations or estimated tag counts to detect when transition occurs within the window and adjust the window size appropriately. WSTD has proved to be relatively more accurate in estimating and distinguishing between periods of drops and when the tag has moved out of the detection range compared to SMURF transition algorithm.

We present an analysis of performance of WSTD compared to SMURF and other fixed window cleaning scheme, using the same experimental scenarios used in SMURF [8]. WSTD is able to adapt its window size to cope with fluctuation of the tag-reader performance due to changes in the environment while relatively accurately detecting the transition points. We first present how WSTD cleans individual tag data and then present how it cleans tag aggregates in the applications which only need to know the number of tags available.

A. Adaptive Individual Tag Cleaning

1) Completeness Requirement

Each epoch is viewed as independent Bernoulli trial (i.e. a sample draw for $tag\ i$) with success probability p_i in (1). This implies that the number of successful observations of $tag\ i$ in the window W_i with w_i epochs (i.e. $W_i = (t - w_i, t]$) is a random variable with a binomial distribution $B(w_i, p_i)$. In the general case, assume that $tag\ i$ is seen only in subset $S_i \subseteq W_i$ of all epochs in the window W_i. Assuming that, the tag probabilities within an approximately sized window calculated using (1), are relatively homogeneous, taking their average will give a valid estimate of the actual probability of $tag\ i$ during window W_i. Therefore, the average empirical read rate p_i^{avg} over the observation epochs is given by (2).

$$p_i^{avg} = (1/|S_i|) \cdot \sum_{t \in S_i} p_{i,t} \quad (2)$$

Also S_i can be seen as a binomial sample of epochs in W_i i.e. a Bernoulli trial with probability p_i^{avg} for success and $|S_i|$ as a binomial random variable with binomial distribution $B(w_i, p_i^{avg})$. Hence, from standard probability theory the expected value and variance of $|S_i|$ is given as

$E[|S_i|] = w_i \cdot p_i^{avg}$ *and*

$Var[|S_i|] = w_i \cdot p_i^{avg} \cdot (1 - p_i^{avg})$ respectively.

The above binomial sampling model is then used to set the window size to ensure that there is enough epochs in the window W_i such that $tag\ i$ is read if it does exist in the reader's range. Setting the number of epochs within the smoothing window using (3) ensures that $tag\ i$ is observed within the window W_i with probability $> 1 - \delta$.

$$w_i \geq \left\lceil (1/p_i^{avg}) ln(1/\delta) \right\rceil \quad (3)$$

2) Adaptive Window Size Adjustment

In order to balance between guaranteeing completeness and capturing tag dynamics the WSTD algorithm uses simple rules together with statistical analysis of the underlying data stream to adaptively adjust the cleaning window size.

Assume $W_i = (t - w_i, t]$ is $tag\ i$ current window, and let $W_{1i}^{'} = (t - w_i, t - w_i/2]$ denote the first half of window W_i and $W_{2i}^{'} = [t - w_i/2, t]$ denote the second half of the window W_i. Let $|S_{1i}|$ and $|S_{2i}|$ denote the binomial sample size during $W_1^{'}$ and $W_2^{'}$ respectively. Note that the mid epoch (i.e. epoch at $t - w_i/2$) is inclusive of both ranges.

Rule 1: Similar to SMURF, transition within the window is detected if the number of observed readings is less than the expected number of readings (i.e. $|s_i| < w_i \cdot p_i^{avg}$) and there is statistically significant variation in the tag observations using the Central Limit Theorem (CLT).

$$\left| |S_i| - w_i p_i^{avg} \right| > 2 \cdot \sqrt{w_i p_i^{avg} (1 - p_i^{avg})}$$

However, we noted that this variation within the window can also be caused by missing tags and not necessarily only due to transition. Hence, to reduce the number of false positives due to transitions and the number of false negative readings which will be further introduced in case of wrong transition detection, the window size is reduced additively by reducing the window size by two epochs.

To improve the transition detection mechanism for the mobile tags we combine the mobile detection mechanism together with the observations of the second half of the window $|S_{2i}|$ to estimate when the tag is exiting the detection range. The slope of the best-fit line using the least squares fitting with the observed probabilities in the window ($\frac{\Delta p_{i,t}}{\text{epochs}}$) is used to determine if the tag is moving out. If the tag is detected with consistently falling $p_{i,t}$, within the window it is inferred that the tag is moving out. Hence, the negative slope of the best-fit line indicates that the tag is moving out.

Rule 2: If the tag is moving out and it was not detected in the second half of the window (i.e. $|S_{2i}| = 0$) the tag is assumed to have exited or exiting the detection range. In this case the window size is halved to reduce the false positive readings. One weakness of this rule is that premature exit transition detection will also lead to false negative readings due to small window sizes.

Rule 3: The window size is increased if the computed window size using (3) is greater than the current window size and the expected number of observation samples is less than the actual number of observed samples (i.e. $|S_i| > w_i p_i^{avg}$). Low expected observation samples indicates that the probability of detection p_i^{avg} is low, in this case we need to grow the window size to give more opportunity for the poor performing tag to be detected. Otherwise if the expected observation sample is equal or greater than the actual sample size it means that, the p_i^{avg} is good enough and we don't have to increase the window size. This rule ensures that the window size is increased only when the read rate is poor.

Figure 1 shows a pseudo-code description of the WSTD adaptive per tag cleaning algorithm. Each individual tag is cleaned in its own window. The rules described above are used to adjust the tag's cleaning window size adaptively based on the statistical analysis of the underlying tag observations. Initially all new detected tag's window are set to 1 epoch, the window sizes are then adjusted according to their detection rates with minimum window size set to 3 epochs. Setting the minimum window size to 3 epochs strikes the balance between maintaining the smoothing effect of the algorithm and reducing the false positive errors. Similar to SMURF, WSTD also slides its window per single epoch (read cycle) and produces output readings corresponding to the midpoint of the window after the entire window has been read.

```
Input:      T = set of all observed tag IDs
            δ = required completeness confidence
Output:     t = set of all present tag IDs
Initialize: ∀i ∈ T, w_i ← 1
while(getNextEpoch) do
    for (i in T)
        processWindow(W_i) → p_{i,t}, p_i^avg, |S_i|
        if ( tagExist(|S_i|)
            output i
        end if
        w_i* ← requiredWindowSize(p_i^avg, δ)
        tagExiting ← mobileDetection(p_{i,t}s,)
        if (tagExiting ∧ |S_2i| = 0)
            w_i ← max (min{w_i/2 , w_i*}, 3)
        else if (detectTransition(|S_i|, w_i, p_i^avg))
            w_i ← max (min{w_i − 2, w_i*}, 3)
        else if (w_i* > w_i ∧ |S_i| < w_i p_i^avg)
            w_i ← min{w_i + 2, w_i*}
        else
            w_i ← min{w_i, w_i*}
        end if
    end for
end while
```

Figure 1: The WSTD Individual tag cleaning algorithm

B. Multi-Tag Aggregate Cleaning

Some applications do not require information for each individual tag, but only need to track simple aggregates (e.g. counts or averages) over the entire tag population. These types of applications typically track large populations of tags. For

instance, a retail store monitoring application may only need to know when the count of items on the shelf or store drop below a certain threshold level. The per tag cleaning method could be used to clean tags in such scenarios, where by each tag in the population is individually cleaned and their result is aggregated across individual smoothing filters for each epoch. This solution can be highly affected by poorly performing tags especially in the static environment. The per tag cleaning algorithm adapts the window size for each individual tag and because window sizes for individual tags might be different based on their detection rates, the decision at whether the tag is present or not is taken at different epochs. Therefore, due to different window sizes, the tags not ready for processing (i.e. the readings for all epochs in its window have not be accumulated) will delay the output. To avoid this limitation caused by low performing tags, multi-tag cleaning algorithm uses the same smoothing window for all tags together with a statistical estimation technique to accurately estimate the tags population count without cleaning on a per-tag basis.

As with individual tag observation, the smoothing window size plays a critical role in capturing the underlying tags population aggregate. A large window ensures that the tags are observed and aggregated with high probability, but a small window is also desired to ensure that variability in the population count is adequately captured.

The multi-tag cleaning mechanism uses some of the concepts proposed in SMURF whereby the Horvitz-Thompson (HT) estimator [19] also known as π-estimator together with unequal-probability random sampling model is used to approximate the population aggregates. As with the per-tag cleaning method, the multi-tag cleaning mechanism also views each epoch as an independent Bernoulli trial with probability p_i^{avg} for success. Where p_i^{avg} denotes the average empirical sampling probability for $tag\ i$ during window W derived from the readers tag list information using (2). The same process as used in the per tag algorithm to determine the size of the window which ensures that tags are read with high probability.

Let S_W denote the sample of distinct tags read over the current smoothing window and let $p^{avg} = (1/|S_W|) \cdot \sum_{i \in S_W} p_i^{avg}$ denote the average per-epoch sampling probability over all observed tags. Following the similar rationale used in the per-tag cleaning, to ensure that the underlying tag population is read with high probability ($\geq 1 - \delta$) we set the upper bound of the smoothing window size for multi-tag aggregate at $w = \lceil (1/p^{avg}) ln(1/\delta) \rceil$. According to binomial distribution, the overall probability of reading $tag\ i$ at least once during window $w = |W|$ is estimated as one minus probability of not detecting $tag\ i$ in all the trials $\pi_i = 1 - \left(1 - p_i^{avg}\right)^w$. Let $S_W \subseteq \{1, \dots, N_W\}$ denote the subset of distinct observed (i.e. sampled) RFID tags over the window W and N_W denote the true tags count. The π-estimator for the population count based on the sample S_W is defined as $\widehat{N_W} = \sum_{i \in S_W} \frac{1}{\pi_i}$. The π-estimator uses the sampling probability π_i to weigh the responses in estimating the population total. The poor performing tags with lower response probability are given higher weights while higher probability responses are given lower weights. The π-estimator gives unbiased estimation of tag population $\widehat{N_W}$ with its estimated mean and variance given as $E\left(\widehat{N_W}\right) = N_W$ and $\widehat{Var}\left(\widehat{N_W}\right) = \sum_{i \in S_W} \frac{1-\pi_i}{\pi_i^2}$ respectively.

1) Adaptive Window Size Adjustment

The WSTD cleaning algorithm employs the random-sampling model and π-estimator concepts proposed in SMURF together with comparison of the two window sub-range estimated tag counts to dynamically adapt its smoothing window size. Transitions are detected as statistically significant changes in aggregate estimates over sub-ranges of its current smoothing window. The transition detection model used is the main difference between our multi-tag cleaning algorithm and the SMURF multi-tag cleaning algorithm.

Assume $W = (t - w, t]$ is current window, and let $W_1' = (t - w, t - w/2]$ denote the first half of window W and $W_2' = [t - w/2, t]$ denote the second half of the window W. Let $\widehat{N_{W_1'}}$ and $\widehat{N_{W_2'}}$ denote the π-estimators for tag population counts during W_1' and W_2' respectively. Note that the mid epoch (i.e. epoch at $t - w/2$) is inclusive in both ranges. The mid-point divides the window such that the numbers of epochs are equally spaced on either side of the window and this requires the use of an odd number window size. The transition is detected if there is significant change in tag counts between these two ranges using the following condition $\left|\widehat{N_{W_1'}} - \widehat{N_{W_2'}}\right| > 2\left(\sqrt{Var\left(\widehat{N_{W_1'}}\right)} + \sqrt{Var\left(\widehat{N_{W_2'}}\right)}\right)$. Our experimental results verified that using the comparison of the sub-range population count estimates to detect population count variation within the window, gives a more accurate transition detection technique than comparison between full window count and the sub-range count estimates used by SMURF. The SMURF detection condition detects any significant variation within the window however for transition detection mechanism we are more interested in detecting the significant changes on the edge of the window which signals that the tag is either entering or leaving the detection range and then respond accordingly.

By comparing the population count of the two window sub ranges, it is possible to determine when the tag is exiting and entering the detection range eliminating the need to use mobile detection algorithm as proposed by SMURF. In the environment where tags are mobile there are two scenarios: one is tags exiting the detection range and the second is tags entering the detection range.

We use simple rules to detect when these transitions occur by comparing the estimated tag count in the two window sub ranges. The tags are said to be exiting the detection range if the transition is detected and there is more estimated tag count in the first half of the window than in the second half of the window (i.e. $\widehat{N_{W_1'}} > \widehat{N_{W_2'}}$). In this case, the window size is reduced multiplicatively (i.e. divided in half) to circumvent false positive readings. Similarly the tag is said to be entering the detection region if transition is detected and there is more

estimated tag count in the second half of the window than in the first half of the window (i.e. $\widehat{N_{W_2'}} > \widehat{N_{W_1'}}$). In this case, if the required window size is greater than twice the current window size, the window size is increased multiplicatively (i.e. doubled) otherwise the window size is additively increased by two epochs. This is because as tag enters the detection range it is assumed to be on the far end of reader's detection ranges i.e. long distance from the reader's antenna. Increasing the window size gives more opportunity even for the weak performing tags to de detected.

To reduce false positive readings during transition period i.e. when the tags are leaving and entering the detection range we made two estimation assumptions. These approximation assumptions are used to detect when the tag(s) completely exit(s) the detection range and when the tag(s) just entered the detection region. In the first assumption, the tag(s) are said to have exited the reader's detection range if the overall window tag count is not zero but the second half of the tag population count is zero (i.e. $\widehat{N_W} > 0 \ \wedge \ \widehat{N_{W_2'}} = 0$). This means that there was no tag observed in the second half of the window $(t - w/2\,, t)$. In the second assumption, the tag(s) are said to have just entered the reader's detection range if the overall window tag count is not zero but the first half of tag population count is zero (i.e. $\widehat{N_W} > 0 \ \wedge \ \widehat{N_{W_1'}} = 0$). This means that there was no tag observed in the first half of the window $(t - w, t - w/2)$.

Considering that the cleaning window size slides by the midpoint, we assume that the observed tags under these scenarios are more likely to be a false positive reading caused by a bigger window size. Therefore, tag(s) observed in these scenarios are dropped and the window size is reduced for exiting scenario and increased appropriately for the entering scenario.

By taking advantage of the π-estimator which scales-up the reading in the window to estimate the underlying tag population we can reduce the window sizes to enhance transition detection, hence the minimum window size can be reduced to 1 epoch. We also introduce another estimation condition which we call *strong region detection*. The aim of *strong region detection* is to detect when the tags within the window are observed with high probability of detection and when there is no significant variation in tag population within the two window sub-ranges.

Let S_i be a binomial sample of epochs in the current window W in which a single tag is observed and p_i^{avg} be the average read rate as defined in the per-tag cleaning approach. Let S_W denote the sample of distinct tags read over the current smoothing window and $p^{avg} = (1/|S_W|) \cdot \sum_{i \in S_W} p_i^{avg}$ denote the average sampling probability over all observed tags and $S^{avg} = (1/|S_W|) \cdot \sum_{i \in S_W} S_i$ denote the average sample of epochs in the window in which the tags where observed. The tags are then said to be observed in the strong detection region if the following condition holds $(\ p^{avg} > (1/W) \cdot S^{avg}) \wedge \left(\left|\widehat{N_{W_1'}} - \widehat{N_{W_2'}}\right| < \left\lceil 0.05 \cdot min\left(\widehat{N_{W_1'}}, \widehat{N_{W_2'}}\right)\right\rceil\right)$. The second portion of the logical condition test if the two window sub-range estimates have relatively small difference of less than 5% of the smallest estimated tag counts. If the condition holds the window size is reduced additively by reducing the window size by two epochs.

Figure 2 shows a pseudo-code description of the adaptive multi tag cleaning algorithm. All the tags are cleaned using the same window. Similar to per-tag cleaning, the smoothing window size is systematically adjusted based on the analysis of the observed tags binomial-sampling data and the transition is detected by comparing the window sub range estimated population counts.

```
Input:    T = set of all observed tag IDs
          δ = required completeness confidence
Output:   t = tags count
Initialize: w ← 1
while(getNextEpoch) do
    for (i in T)
      processWindow(W) → p_{i,t}, |S_i|, p_i^avg, p^avg, S^avg, N̂_W, N̂_{W1'}, N̂_{W2'}
    end for
    W* ← requiredWindowSize(P^avg, δ)
    transition ← |N̂_{W1'} − N̂_{W2'}| > 2(√Var(N̂_{W1'}) + √Var(N̂_{W2'}))
    exitTransition ← transitionTest ∧ N̂_{W1'} > N̂_{W2'}
    enterTransition ← transitionTest ∧ N̂_{W2'} > N̂_{W1'}
    exit ← N̂_W > 0 ∧ N̂_{W2'} == 0
    enter ← N̂_W > 0 ∧ N̂_{W1'} == 0
    strongDetection ← ( p^avg > (S^avg/W)) ∧ (|N̂_{W1'} − N̂_{W2'}| <
                      ⌈0.05 · min(N̂_{W1'}, N̂_{W2'})⌉)
    if (exit ∨ exitTransition ∨ strongDetection)
        if ( exit)
            t = 0
        else
            t = N̂_W
        end if
        if ( strongDetection)
            W ← max (min(W − 2, W*) , 1)
        else
            W ← max (min (W/2, W*), 1)
        end if
    else if ((w_i* > w_i) ∧ (|S^avg| < w · p^avg ))
        if ( enter)
            t = 0
        else
            t = N̂_W
        end if
        if (W* > 2 * W ∧ (enter ∨ enterTransition))
            W ← min (W * 2, W*)
        else
            W ← min (W + 2, W*)
        end if
    else
        t = N̂_W
        W ← min (W, W*)
    end if
    output (t)
end while
```

Figure 2: WSTD-pi Multi-tag cleaning algorithm

V. EXPERIMENTAL EVALUATION OF WSTD

In this section we present the experimental evaluation of our proposed WSTD cleaning algorithms. The data sets for our experiments were generated by a synthetic data generator that simulates the operation of RFID readers under a wide

variety of conditions using similar models proposed in SMURF and MATLAB. In our experiments we used a maximum read rate of 95% which is a read rate within the strong-in-field region and completeness confidence δ set to 0.01. Maximum reader detection range of 4.6 m (~15 feet) and varied the strong-in-field percentage ($StrongPercentage$) and the distance between the tag and the reader. In the individual tag cleaning algorithms we used 25 tags while in the multi-tag aggregate algorithms we used 100 tags and the data were generated for 2000 read cycles (epochs).

We compare the performance effectiveness of the WSTD algorithms with that of SMURF and other fixed window-based cleaning methods using the generated synthetic data sets. We denote each fixed window method as Fxd$x$, where x is the size of static cleaning window in epochs.

A. Individual Tag Cleaning

1) Experiment1: Environment Reliability with the randomly moving tags

In this experiment we determine how each technique reacts to different levels of environment unreliability with the randomly moving tags. Each tag is moved with its own random velocity between 0 to 90 cm/epoch and after every 100 epochs on average the tag change its state from moving to rest state and vice versa. When the tag resumes movement it chooses another random velocity, this movement pattern is referred as *Fido* in [8]. The strong-in-field region percentage is varied between 0 and 100%. The lower $StrongPercentage$ corresponds to unreliable environment and higher values of $StrongPercentage$ corresponds to a more controlled environment. At each $StrongPercentage$ we measure the average errors produced by each scheme. The average error per epoch is calculated as

$$\sum_{i=1}^{NumEpochs}(FalsePositive_i + FalseNegative_i)/NumEpochs.$$

Figure 3 shows the result of this experiment.

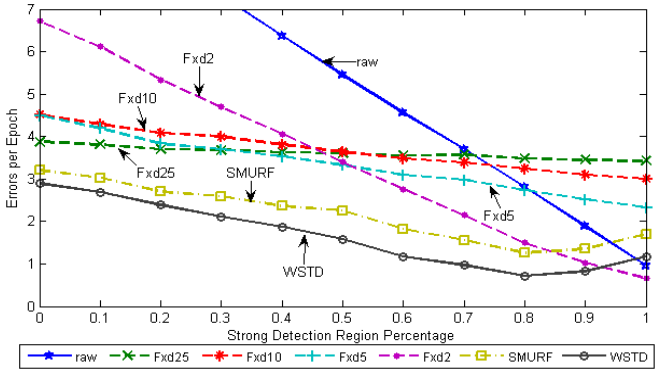

Figure 3: Average errors per epoch as strong-in-field region percentage is varied

In general all schemes have the same pattern whereby their performance improves as the environment noise decreases and the reader produces more reliable data. However the rate of improvement is inversely proportional to the window size, with small windows having high improvement rate than big windows. Looking at the fixed window schemes, there is no single fixed window scheme which performs consistently well than other schemes in all the environments. In the noisy environment large windows perform well than small window schemes while in controlled environment small windows perform better than large windows. In mobile environment, as reader data becomes more reliable big window sizes although efficiently reduces the false negative errors but introduces more false positive errors causing the overall error to be higher than the raw error.

On the other hand variable window schemes SMURF and WSTD perform consistently well across the entire range of environments. Its performance efficiency is attributed to its per tag cleaning concept whereby each tag's smoothing window is adjusted independently based on its individual random behavior. The WSTD scheme performs better than SMURF producing an improvement of about 25% less overall error than that produced by SMURF. This performance is attributed to its improved transition detection mechanism as shown in Figure 4. Comparing the two variable cleaning window sizes, WSTD uses smaller window size than that used by SMURF as shown in Figure 5.

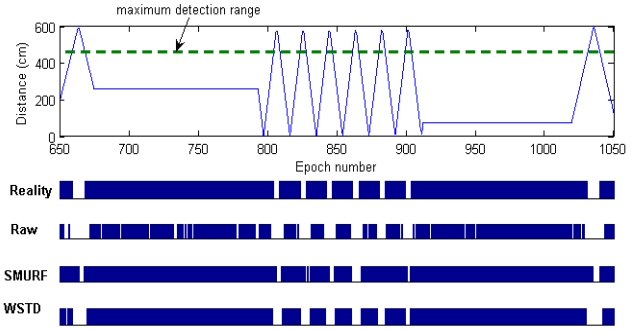

Figure 4: Comparison of WSTD and SMURF schemes transition detection mechanisms as a tag moves at random velocity

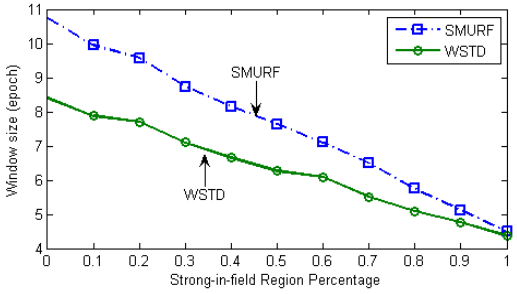

Figure 5: Comparison of WSTD and SMURF schemes cleaning window sizes as the environment noise is varied

Because of its small window size WSTD is more efficient in detecting transition than SMURF however it also produces slightly more negative errors than SMURF as shown in Figure 6. The increase in false negative errors in the noisy environment by WSTD can be associated with the premature transition detection by *rule2* of the WSTD algorithm. As the noise decreases, their performance in compensating for missed readings becomes competitive and their difference decreases.

2) Experiment 2: Effect of tag speed

The effectiveness of the individual tag cleaning schemes is then compared as the tag velocity is varied. The

StrongPercentage parameter is fixed at 70% to represent the controlled environment and the tags are moved in and out of the detection range at the same constant velocity, this motion as referred as *Pallet* in [8]. The velocity is varied from 0 to 90 cm/epoch and we measure the average errors produced by each scheme. Figure 7 shows the result of this experiment.

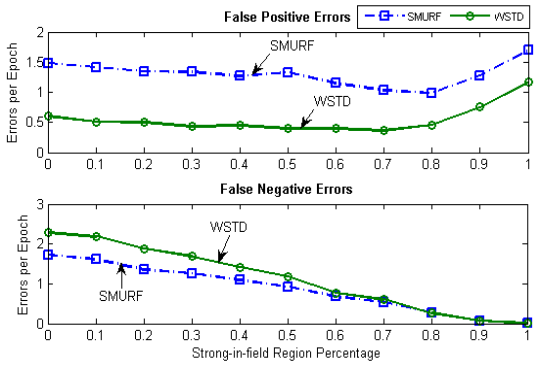

Figure 6: WSTD and SMURF schemes false positive and false negative error contributions as the environment noise is varied

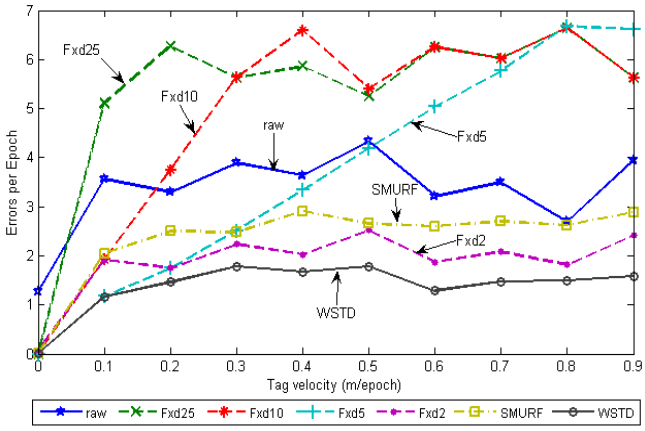

Figure 7: Average errors per epoch as tag velocity varies

In the mobile environment, the larger the window size, the higher the false positive errors. At a one particular fixed window size the false positive errors increases with the increase in tag speed until it reaches a saturation speed beyond which no transition is detected. Beyond the saturation speed in the worst case scenario the scheme continuously reports all tags as being present with no false negatives. The saturation speed increases with the decrease in window size that is the higher the window size the lower the saturation speed e.g. in Figure 7, 'Fxd25', 'Fxd10' and 'Fxd5' have saturation speeds of 0.1, 0.3 and 0.8 m/epoch respectively.

The small window scheme ('Fxd2') has a relatively consistent performance irrespective of the change of the velocity. This is because small windows are able to detect transitions caused by varying velocities although they are unable to compensate for the missing tags. On the other hand, variable window schemes SMURF and WSTD perform consistently well and also outperforms the 'Fxd2' scheme, this is because in addition to being able to detect transition they are also able to compensate for missed readings. The WSTD scheme performs better than SMURF producing an improvement of about 30% less overall error than that produced by SMURF. This performance improvement is attributed to its improved transition detection as shown in Figure 8 due to its use of smaller window sizes. Figure 9 shows the false positive and false negative error contributions of the two variable window schemes.

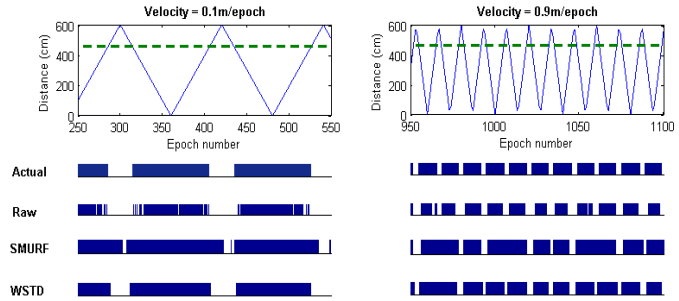

Figure 8: Comparison of WSTD and SMURF schemes transition detection mechanisms as a tag moves at constant velocity

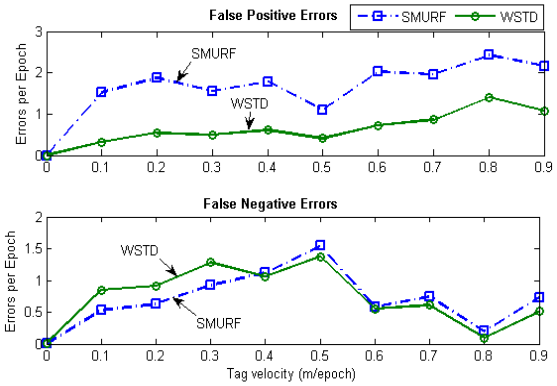

Figure 9: Comparison of WSTD and SMURF schemes false positive and false negative error contributions as the velocity is varied

B. Tag Aggregation Cleaning

1) Experiment 3: Effect of tag speed on aggeregate tag cleaning

We evaluate the tag count accuracy of the cleaning schemes as the tags velocity is varied. The *StrongPercentage* parameter is fixed at 25% to represent the noisy environment and the tags are moved in and out of the detection range at the same velocity. The velocity is varied from 0 to 90 cm/epoch and we measure the RMS errors produced by each scheme. Figure 10 shows the result of this experiment. These results reveal the same fact, that large static windows are not ideal in cleaning data in the mobile tags environment. The large fixed window schemes beyond saturation speed are unable to detect transition and constantly report all the tags as being present leading to high number of overestimated tag count. Smaller fixed windows are unable to compensate for missed readings and constantly produce under count results. On the other hand, variable window schemes SMURF, WSTD and WSTD-π perform better than the fixed window schemes. WSTD-π outperforms all the other schemes producing relatively stable and lower errors than all the other schemes compared. Its performance is contributed to its ability to detect transition and

adjust the window accordingly and the use of π-estimator to estimate the number of tags.

Figure 11 shows the comparison of the reported estimated tags count and that of the actual tag count for the three variable window schemes SMURF, WSTD and WSTD-π with the tags moving at velocity of 0.4m/epoch in the noisy environment. WSTD-π provides closely accurate tag count estimation compared to other compared schemes.

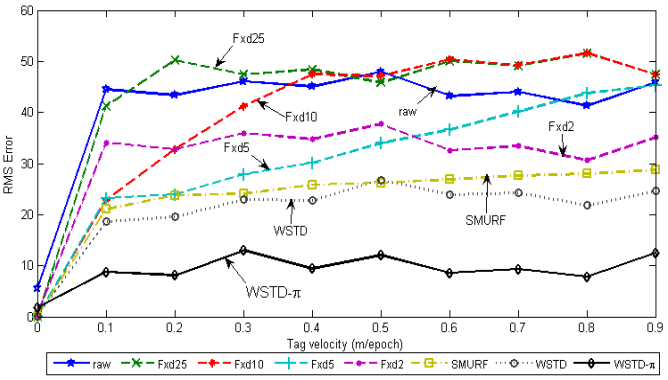

Figure 10: The RMS error of different cleaning schemes counting 100 tags as their velocities varies from 0 to 0.9m/epoch in the noisy environment with ***StrongPercentage*** parameter set to 25%

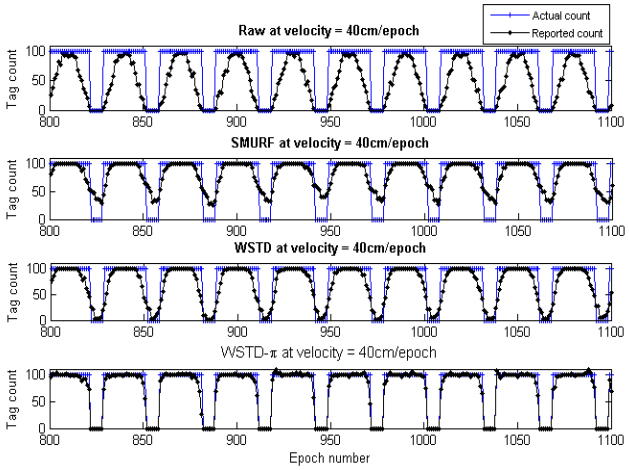

Figure 11: Comparison of variable window cleaning schemes reported tags count with the actual tag count with ***StrongPercentage*** parameter set to 25%.

VI. Conclusions and Future Work

In this paper we have studied the RFID missing readings problem and the sliding window based approaches which are used to address these problems. We looked at the challenges associated with setting the appropriate sliding window size especially in the mobile tag environments. We developed our adaptive sliding window based cleaning scheme for RFID data stream with an improved transition detection mechanism. Our scheme WSTD uses binomial sampling concepts to calculate the appropriate window size and π-estimator to estimate the number of tags, and then uses the comparison of the two window sub-range observations or estimated tag counts to detect when transition occurs within the window.

In the mobile environment WSTD scheme performs better than SMURF producing an improvement of about 30% less overall errors compared with SMURF. This performance improvement is attributed to its improved transition detection mechanism. WSTD uses smaller window sizes than SMURF which means that WSTD also takes shorter processing time compared to SMURF. This work is part of our ongoing work on developing multi-agent based RFID middleware system. Our future work will focus on the ways to deal with duplicate and false positive readings.

References

[1] K. Ahsan, H.Shah and P.Kingston, "RFID applications: An introductory and exploratory study," IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 1, No. 3, January 2010.

[2] A. Mitrokotsa, C. Douligeris, "Integrated RFID and sensor networks: Architectures and applications," Y. Zhang, L.T. Yang, J. Chen, Eds. *RFID and Sensor Networks*, CRC Press, 2009, pp.511-535.

[3] EPCglobal Inc. EPC Radio Frequency Identification protocols class-1 generation-2 UHF RFID protocol for communications at 860mhz–960mhz, Standard Specification version 1.2.0.

[4] M. Boli´c, A. Athalye, and T. Hao Li. Performance of Passive UHF RFID Systems in Practice. In Book RFID Systems: Research Trends and Challenges. Editors M. Boli'c, D. Simplot-Ryl and I. Stojmenovi´c. John Wiley & Sons Ltd, 2010.

[5] M. Buettner and D. Wetherall. "An empirical study of UHF RFID systems," in *MobiCom'08*, San Francisco, California, USA, 2008.

[6] S. Aroor and D. Deavours. "Evaluation of the state of passive uhf rfid: An experimental approach," in *IEEE Systems Journal*, volume 1, 2007 pp.168–176.

[7] Y. Kawakita and J. Mitsugi, "Anti-collision performance of Gen2 Air Protocol in Random error Communication Link," in Proceedings of the International Symposium on Applications and Internet Workshops (SAINT'06), 2006, pp.68-71.

[8] S.R. Jeffery, M. Garofalakis, and M.J.Franklin, "Adaptive cleaning for RFID data streams," *Proceedings of the 32nd international conference on Very large data bases*, VLDB Endowment, 2006, pp 163-174.

[9] L. V. Massawe, F. Aghdasi and J. Kinyua, "The Development of a Multi-Agent Based Middleware for RFID Asset Management System Using the PASSI Methodology.", IEEE Computer Society: Proceedings of the 6th ITNG conference, 2009, pp.1042-1048.

[10] P. Darcy, B.Stantic and A.Sattar, "A Fusion of Data Analysis and Non-Monotonic Reasoning to Restore Missed RFID Readings," in Proceedings of Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2009), 2009, pp.313-318.

[11] M. S. Trotter and G. D. Durgin, "Survey of Range Improvement of Commercial RFID Tags with Power Optimized Waveforms," in IEEE International Conference on RFID, April 2010, 195-202.

[12] A. Rahmati, L. Zhong, M. Hiltunen, and R. Jana, "Reliability Techniques for RFID-Based Object Tracking Applications," in Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '07). IEEE Computer Society, 2007, pp.113–118.

[13] H. Chen, W. Ku, H. Wang and M. Sun, " Leveraging spatio-temporal redundancy for RFID data cleansing," In Proceedings of the 2010 international conference on Management of data, SIGMOD '10, 2010

[14] H. Gonzalez, J. Han, and X. Shen, "Cost-conscious cleaning of massive RFID data sets," in Proc. 2007 Int. Conf. on Data Engineering (ICDE'06), Istanbul, Turkey, April 2007.

[15] B. Song, P. Qin, H. Wang, W. Xuan, G. Yu, "bSpace: A data cleaning approach for RFID data streams based on virtual spatial granularity," in Proceedings of HIS (3), 2009, pp.252-256.

[16] J. Rao, S. Doraiswamy, H. Thakkar and L. S. Colby, "A Deferred Cleansing Method for RFID Data Analytics," in VLDB, 2006.

[17] A. Gupta and M. Srivastava, "Developing Auto-ID solutions using Sun Java system RFID software," Oct 2004.

[18] C. Bornhoevd, T. Lin, S. Haller, and J. Schaper, "Integrating automatic data acquisition with business processes - Experiences with SAP's Auto-ID infrastructure," presented at 30th international conference on very large data bases (VLDB), Toronto, Canada, 2004.

[19] S. L. Lohr. *Sampling: Design and analysis.* New York: Duxbury Press (1999).